

PROGRAMMING I

Curriculum Content Frameworks

Prepared by

Xerlotta Sanders, Central High School, Little Rock
Wayne Martin, Shirley High School
Marilyn Carrell, Springdale High School

Facilitated by

Karen Chisholm, Program Manager
Office of Assessment and Curriculum
Arkansas Department of Workforce Education

Edited by

Linda Shock, Program Manager
Doretta Griffin, Public School Program Advisor
Brenda Buerkle, Public School Program Advisor
Jim Brock, Public School Program Advisor
Ted Dean, Public School Program Advisor
Office of Business/Marketing Technology
Arkansas Department of Workforce Education

Disseminated by

Career and Technical Education
Office of Assessment and Curriculum
Arkansas Department of Workforce Education

Programming I

The experience of the members of the frameworks committee, who have taught programming classes for several years, is that Algebra I be a strongly enforced prerequisite. It is our experience that students who have not successfully completed Algebra I (with an A or B) will not be successful in this class. Not only are mathematical concepts and formulas incorporated into the skills, but the problem-solving skills necessary are not adequately developed until the completion of Algebra I.

Depending on the language used, the terms *functions*, *subprogram*, *method*, and *procedure* are similar. However, they are used somewhat differently. In some languages, the terms *subprogram* and *procedure* have the same effect as a void function in C++ and Java. In those languages, the term *function* is used only to apply to functions that return a value (in C++ and Java, all of these are called functions).

The contents of these frameworks are not intended to be taught in this order as independent units. Many of the skills are best introduced in one unit and then spiraled back to in future units with more complexity added. However, by the end of the semester, all skills should be taught.

The contents of these frameworks are designed to be taught in one language. The first semester of any language should be Programming I. Currently, almost certainly the three best language choices are Visual Basic, Java, and C++. Java has an advantage over C++ since the College Board has selected it for the language of the Advanced Placement Exam, and many universities are using Java as their first programming language. Visual Basic is widely used in business programming.

The contents of these frameworks are kept to the essentials. This was done to allow the teacher time to address the specific features of the language chosen. The framework team recognized there are vastly different additional items that need to be addressed in a visual Windows application (like Visual Basic) rather than in a console application (like those used by the College Board). We expect the teacher will use the remaining time in the semester to cover those topics not listed in these frameworks.

Curriculum Content Frameworks

PROGRAMMING I

Grade Levels: 9, 10, 11, 12
Course Code: 492390

Prerequisite: Keyboarding and Algebra I
(Strongly Recommended)

Course Description: Programming I is a one-semester course in any modern, high-level, structured language. Concepts should be taught in the context of practical applications.

Table of Contents

	Page
Unit 1: Introduction to Programming and Ethics in Programming	1
Unit 2: Programming Techniques and Characteristics of Good Programs	3
Unit 3: Data Types and Mathematical Operations	5
Unit 4: Printing and Formatting	7
Unit 5: Structured and Object-oriented Programming	8
Unit 6: Interactive Program, Program Execution, Prompt	9
Unit 7: Decision Structure	10
Unit 8: Loops	11
Glossary	13

Unit 1: Introduction to Programming and Ethics in Programming

Hours: 3

Terminology: Application software, Compiler, Hardware, High-level language, Interpreter, Low-level language, Operating system, Software, System software

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
1.1 Explain the difference between system and application software	1.1.1 Identify various software as system or application	Foundation	Speaking	Communicates a thought, idea, or fact in spoken form [1.5.5]	
		Thinking	Knowing how to Learn	Applies new knowledge and skills to differentiate between system and application software [4.3.1]	
1.2 Define terms related to hardware and software	1.2.1 Identify technology as either hardware or software	Foundation	Reading	Understands technical words that pertain to hardware/software [1.3.6]	
		Thinking	Knowing how to Learn	Applies new knowledge and skills to differentiate between hardware/software [4.3.1]	
1.3 Discuss various operating systems and their differences (i.e., Windows, Mac, Linux)	1.3.1 Tell where each operating system is used most frequently	Foundation	Reading	Understands technical words that pertain to various operating systems and their differences [1.3.6]	
		Thinking	Reasoning	Sees relationship among various operating systems [4.5.5]	
1.4 Explain the difference between high-level and low-level languages	1.4.1 Classify commonly used programming languages as high-level or low-level	Foundation	Reading	Understands technical words that pertain to high- and low-level languages [1.3.6]	
		Thinking	Reasoning	Sees relationship between high- and low-level languages [4.5.5]	
1.5 Explain the difference between interpreters and compilers	1.5.1 Give an example of how a compiler functions verses how an interpreter functions	Foundation	Speaking	Applies/Understands technical words that pertain to subject [1.3.6]	
		Thinking	Reasoning	Sees relationship between interpreters and compilers [4.5.5]	

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS		
What the Student Should be Able to Do		What the Instruction Should Reinforce		
Knowledge	Application	Skill Group	Skill	Description
1.6 Discuss the ethical and privacy issues of programming	1.6.1 Identify ethical and privacy practices in computer programming	Foundation	Speaking	Applies/Uses technical terms as appropriate to audience [1.5.2]
		Thinking	Reasoning	Sees relationship between ethics/privacy issues and programming [4.5.5]

Unit 2: Programming Techniques and Characteristics of Good Programs

Hours: 2

(Reinforced Throughout the Semester)

Terminology: Algorithm, Documentation, Logic error, Program maintenance, Pseudocode, Run-time error, Syntax, Syntax error, User-friendly

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS				
What the Student Should be Able to Do		What the Instruction Should Reinforce				
Knowledge		Application		Skill Group	Skill	Description
2.1	List the steps of the programming process	2.1.1	When given an example, be able to identify the correct steps	Foundation	Science	Solves practical problems using scientific methods and techniques [1.4.22]
					Writing	Organizes information in an appropriate format [1.6.10]
				Thinking	Reasoning	Sees relationship between steps in the programming process [4.5.5]
2.2	Identify the syntax to document programs	2.2.1	Use appropriate syntax to include comments in programs	Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]
				Thinking	Knowing how to Learn	Applies new knowledge and skills to properly document programs [4.3.1]
2.3	Explain the characteristics of user-friendly programs	2.3.1	Write programs that have clear instructions	Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]
		2.3.2	Write programs whose output is easy to read and understand	Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.3.1]
2.4	Explain the importance of program documentation and maintenance	2.4.1	Write programs that are well-documented	Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]
		2.4.2	Update an existing program	Personal Management	Organizational Effectiveness	Comprehends the organization's modes of operation [3.3.5]
				Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]

CAREER and TECHNICAL SKILLS			ACADEMIC and WORKPLACE SKILLS		
What the Student Should be Able to Do			What the Instruction Should Reinforce		
Knowledge	Application		Skill Group	Skill	Description
2.5 Explain the importance of program documentation and maintenance	2.5.1	Write a pseudopodia (algorithm) for a programming problem	Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]
			Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
2.6 Identify different types of errors	2.6.1	When given an example, identify the error type	Thinking	Knowing how to Learn	Applies new knowledge and skills properly [4.3.1]
2.7 Explain the characteristics of readable programs	2.7.1	Use indentation and blank space to make a program more readable	Foundation	Writing	Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]
	2.7.2	Use descriptive identifiers	Thinking	Knowing how to Learn	Applies new knowledge and skills properly [4.3.1]
	2.7.3	Document difficult logic to make it easy to follow			

Unit 3: Data Types and Mathematical Operations

Hours: 15

Terminology: Boolean, Constant, Data type, Floating point (real), Integer, Mathematical operators, Order of operations, Random numbers, Round-off error, String, Variable

CAREER and TECHNICAL SKILLS			ACADEMIC and WORKPLACE SKILLS		
What the Student Should be Able to Do			What the Instruction Should Reinforce		
Knowledge	Application		Skill Group	Skill	Description
3.1 Discuss string, integer, Boolean, and floating-point	3.1.1 Compare the four data types		Foundation	Arithmetic/ Mathematics	Comprehends mathematical ideas and concepts related to the characteristics of numeric data [1.1.13]
	3.1.2 Determine whether an identification number (such as a Social Security Number) should be treated as a string or			Writing	Presents answers/conclusions in a clear and understandable form [1.6.13]
	3.1.3 Determine whether a particular "number" would be considered numeric		Thinking	Reasoning	Comprehends ideas and concepts related to data [4.5.2]
	3.1.4 Determine whether a number should be treated as an integer or floating point (i.e., single, double)				
	3.1.5 Designate data type using correct syntax				
3.2 Explain the advantages of using integer variables whenever possible (faster computation, require less memory, obtain exact answers)	3.2.1 Use integer variables in programs where appropriate		Foundation	Arithmetic/ Mathematics	Comprehends mathematical ideas and concepts related to integer variables [1.1.13]
			Thinking	Decision Making	Comprehends ideas and concepts related to integer variables [4.2.2]
3.3 Explain the advantage and disadvantages of floating-point numbers (round-off errors, more money, approximate answers, slower computation, size of numbers to be stored, etc.)	3.3.1 Use floating-point variables in programs where appropriate		Foundation	Arithmetic/ Mathematics	Comprehends mathematical ideas and concepts related to floating-point variables [1.1.13]
				Writing	Uses words appropriately [1.6.21]
				Decision Making	Comprehends ideas and concepts related to floating-point numbers [4.2.2]
3.4 Explain the characteristics of string data	3.4.1 Write programs that contain strings		Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise matter [1.6.6]
			Thinking	Decision Making	Comprehends ideas and concepts related to the characteristics of string data [4.2.2]

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS			
What the Student Should be Able to Do		What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
3.5 List arithmetic operations and order of operations	3.5.1 Write formulas using operators and order of operations	Foundation	Arithmetic/ Mathematics	Comprehends mathematical ideas and concepts related to arithmetic operations/ order of operations [1.1.13]	
	3.5.2 Write programs that use mathematical operations correctly	Thinking	Reasoning	Comprehends ideas and concepts related to programs with mathematical operations [4.5.2]	
3.6 Explain rules for choosing variable names	3.6.1 Write programs that use descriptive variable names	Foundation	Writing	Uses words appropriately [1.6.2]	
	3.6.2 Select legal variable names	Thinking	Reasoning	Comprehends ideas and concepts related to variable names [4.5.2]	

Unit 4: Printing and Formatting

Hours: 5

Terminology: Formatting

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
4.1 Explain the syntax to print variables and answers to mathematical operations	4.1.1 Write programs that print the contents of variables and the results of mathematical operations	Foundation	Writing	Presents answers/conclusions in a clear and understandable form [1.6.13]	
		Thinking	Reasoning	Comprehends ideas and concepts related to printing variables and results of mathematical operations [4.5.2]	
4.2 Explain the syntax to properly space output on a line	4.2.1 Write programs that output sentences with string and numeric data	Foundation	Writing	Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]	
		Thinking	Reasoning	Comprehends ideas and concepts related to formatting [4.5.2]	
4.3 Explain the syntax for formatting numeric data	4.3.1 Print numbers with a specific number of decimal places	Foundation	Writing	Organizes information in an appropriate format [1.6.10]	
	4.3.2 Print numbers in currency	Thinking	Reasoning	Comprehends ideas and concepts related to formatting [4.5.2]	
4.4 Explain the commands and syntax for editing numeric and string data	4.4.1 Print numeric data with a fixed number of digits after the decimal point	Foundation	Writing	Applies/Uses technical words and concepts [1.6.4]	
	4.4.2 Print numeric data with commas			Organizes information in an appropriate format [1.6.10]	
	4.4.3 Print data using a fixed length	Thinking	Reasoning	Comprehends ideas and concepts related to formatting [4.5.2]	

Unit 5: Structured and Object-oriented Programming

Hours: 10

Terminology: Function, Hierarchy chart, Method, OOP (Object-oriented programming), Procedure, Structure chart, Structured programming, Subprogram, Visual table of contents (VTOC)

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do			ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce		
Knowledge	Application		Skill Group	Skill	Description
5.1 Define and explain structured programming (programs with main programs and functions, procedures, or subs)	5.1.1 When shown an example, determine whether a program is structured		Foundation	Speaking	Communicates a thought, idea, or fact in spoken form [1.5.5]
			Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
5.2 Define and explain the function of a hierarchy chart, visual table of contents (VTOC), or structure chart	5.2.1 Draw a hierarchy chart, VTOC, or structure chart for structured programs		Foundation	Writing	Applies/Uses technical words and concepts [1.6.4] Composes and creates documents -- letters, manuals, reports, proposals, graphs, flow charts, etc. [1.6.8]
			Thinking	Reasoning	Sees relationship between two or more ideas, objects, or situations [4.5.5]
5.3 Describe the purpose of the main module, and list the rules for designing submodules	5.3.1 Write programs that use the main programs as a control module		Foundation	Writing	Organizes information in an appropriate format [1.6.10]
	5.3.2 Write complete structured programs		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
5.4 Define object-oriented programming (OOP)	5.4.1 List a language that uses OOP		Foundation	Speaking	Communicates a thought, idea, or fact in spoken form [1.5.5]
			Thinking	Reasoning	Comprehends ideas and concepts related to OOP [4.5.2]
5.5 Explain the use of dot notation in OOP	5.5.1 Write programs that use dot notation to use the methods in classes that are part of the language (i.e., textbox.text in VB or system.out in Java)		Foundation	Writing	Organizes information in an appropriate format [1.6.10]
			Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]

Unit 6: Interactive Program, Program Execution, Prompt

Hours: 10

Terminology: Interactive program, Program execution, Prompt

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
6.1 Discuss interactive programs	6.1.1 Write a program that gets data from a user during execution (visual languages should use both text boxes and input boxes)	Foundation	Speaking	Communicates thoughts, ideas, or facts in spoken form [1.5.5]	
		Thinking	Reasoning	Comprehends ideas and concepts related to interactive programs [4.5.2]	
6.2 Describe the qualities of good prompts	6.2.1 Write programs that use good prompts	Foundation	Writing	Communicates thoughts, ideas, or facts in written form in a clear, concise manner [1.6.6]	
		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	
6.3 Identify the syntax for obtaining data from a keyboard during program execution	6.3.1 Write programs that get data from the keyboard during program execution	Foundation	Writing	Applies/Uses technical words and concepts [1.6.4]	
		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	

Unit 7: Decision Structure

Hours: 15

Terminology: Boolean expression, Logical operators, Nested statement, Relational operator, Selection structure (switch/select case), Truth table

CAREER and TECHNICAL SKILLS		ACADEMIC and WORKPLACE SKILLS				
What the Student Should be Able to Do		What the Instruction Should Reinforce				
Knowledge		Application		Skill Group	Skill	Description
7.1	List relational operators	7.1.1	Use the appropriate relational operator	Foundation	Arithmetic/ Mathematics	Interprets mathematical symbols [1.1.26]
		7.1.2	Determine the relationship when used with string data		Writing	Applies/Uses technical words and concepts [1.6.4]
				Thinking	Reasoning	Comprehends ideas and concepts related to relational operators [4.5.2]
7.2	Explain the syntax and logic of If statements	7.2.1	Write programs that use block If-Then statements	Foundation	Writing	Organizes information in an appropriate format [1.6.10]
				Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
7.3	Explain the syntax and logic of If-Then-Else statements	7.3.1	Write programs that use block If-Then-Else statements	Foundation	Writing	Organizes information in an appropriate format [1.6.10]
				Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
7.4	Explain the syntax and logic of nested statements	7.4.1	Write programs using block If-Then-Else for three or more alternatives	Foundation	Writing	Organizes information in an appropriate format [1.6.10]
				Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]
7.5	Explain the use of logical operators <i>and</i> , <i>or</i> , <i>not</i>	7.5.1	Write programs that require the use of <i>and</i> , <i>or</i> , <i>not</i>	Thinking	Reasoning	Sees relationship between two or more ideas, objects, or situations [4.5.5]
7.6	Use selection structure (select case, switch case)	7.6.1	Write programs that use the selection structure of the language	Foundation	Writing	Organizes information in an appropriate format [1.6.10]
				Thinking	Problem Solving	Devises and implements a plan of action of resolve problems [4.4.3]

Unit 8: Loops

Hours: 20

Terminology: Accumulator, Counter, Nested loop, Sentinel loop

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce			
Knowledge	Application	Skill Group	Skill	Description	
8.1 Explain <i>for</i> loops	8.1.1 Write programs that use simple counting loops	Foundation	Writing	Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]	
		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	
8.2 Explain the procedure to use loops to count in increments/decrements other than one	8.2.1 Write counting for loops with increments other than one	Foundation	Writing	Applies/Uses technical words and concepts [1.6.4] Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]	
		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	
8.3 Explain the syntax of nested loops	8.3.1 Determine the output of a nested loop	Foundation	Writing	Applies/Uses technical words and concepts [1.6.4]	
	8.3.2 Write programs using nested loops			Organizes information in an appropriate format [1.6.10] Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]	
		Thinking	Decision Making	Demonstrates decision-making skills [4.2.4]	
			Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]	

CAREER and TECHNICAL SKILLS What the Student Should be Able to Do		ACADEMIC and WORKPLACE SKILLS What the Instruction Should Reinforce		
Knowledge	Application	Skill Group	Skill	Description
8.4 Explain the logic of <i>do while</i> loops	8.4.1 Write programs with <i>do while</i> loops	Foundation	Writing	Applies/Uses technical words and concepts [1.6.4] Uses language, style, organization, and format appropriate to subject matter, purpose, and audience [1.6.19]
		Thinking	Problem Solving	Devises and implements a plan of action to resolve problems [4.4.3]

Glossary

Unit 1: Introduction to Programming and Ethics in Programming

1. Application software – software that is used to accomplish a specialized task; i.e., word processing, spreadsheets, games, instructional programs, tax software, and graphics programs
2. Compiler – a computer program that translates the code of a high-level language into machine code and stores that machine-readable code in an executable file
3. Hardware – the physical components that make up a computer system
4. High-level language – a programming language that uses English-like words and symbols
5. Interpreter – a program that translates a high-level language into machine code one line at a time as the program is being executed
6. Low-level language – a programming language that is non-English-like, such as machine code or assembly language
7. Operating system – the software that allows the computer to communicate with the peripherals and that manages or controls the function of the computer
8. Software – computer instructions that cause the computer to perform desired tasks
9. System software – programs that allow users to write and execute other programs

Unit 2: Programming Techniques and Characteristics of Good Programs

1. Algorithm – a set of steps to follow to solve a problem
2. Documentation – explanation of how a program works; i.e., comments inside the program, flow charts, and other explanatory material
3. Logic error – an error in a program that will run, which produces incorrect, unexpected results
4. Program maintenance – the process of keeping a program up-to-date and edited to match changing conditions
5. Pseudocode ("fake program") – a set of instructions to solve a programming problem in English or a combination of English and the programming language
6. Run-time error – an error that is detected after a program is running; produces an error message rather than an incorrect answer like a logic error would
7. Syntax – the grammar rules (spelling, punctuation, etc.) of a language
8. Syntax error – a grammar error; spelling, punctuation, etc.
9. User-friendly – a program that is easy to use with clear, easy-to-understand instructions

Unit 3: Data Types and Mathematical Operations

1. Boolean – a data type that can store only false or true
2. Constant – a named memory cell that contains a value that cannot be changed as the program runs
3. Data type – a way to specify what kind of data can be stored in a variable or constant; i.e., integer, double, single, Boolean, char, string
4. Floating point (real) – a method of writing numbers in scientific notation to accommodate numbers that may be very large or very small; these numbers may include a decimal point; i.e., float, single, and double
5. Integer – a data type that can store whole numbers, both positive and negative
6. Mathematical operators – symbols that represent mathematical actions (+, -, *, /, etc.)
7. Order of operations – the order in which mathematical operations are performed: parenthesis, exponents, multiplication/division, addition/subtraction
8. Random numbers – numbers that are produced in a nonpatterned manner
9. Round-off error – a condition in which a portion of a real number is lost because of the way it is stored in memory
10. String – a group of characters enclosed in quotation marks
11. Variable – a named memory location that contains values that can be changed as the program is executed

Unit 4: Printing and Formatting

1. Formatting – designating the way data should appear; includes displaying a dollar sign, a specified number of digits, the percent sign with numbers

Unit 5: Structured and Object-oriented Programming

1. Function – a section of named code that performs a specific task
2. Hierarchy chart – a graphical method of showing the relationship between the modules of the program; called *structure chart* or *VTOC* in some textbooks
3. Method – a section of named code that performs a specific task
4. Object-oriented programming (OOP) – building a program by creating, controlling, and modifying objects; defining a class with a set of data with methods to manipulate and use that data
5. Procedure – a section of named code that performs a specific task
6. Structure chart – a graphical method of showing the relationship between the modules of the program; called *hierarchy chart* or *VTOC* in some textbooks
7. Structured programming – a program that implements a top-down design in which the code is broken up into modules
8. Subprogram – a section of named code that performs a specific task
9. Visual table of contents (VTOC) – a graphical method of showing the relationship between the modules of the program; called *structure chart* or *hierarchy chart* in some textbooks

Unit 6: Interactive Program, Program Execution, Prompt

1. Interactive program – a program that gets information from and provides responses to users
2. Program execution – the carrying out of the instructions of the program; running the program
3. Prompt – the question or instructions that ask the user for input

Unit 7: Decision Structure

1. Boolean expression – an expression that evaluates to either true or false; i.e., $\text{age} > 15$, which is either false or true
2. Logical operators – *and*, *not*, *or*; the symbols that represent *and*, *not*, *or*
3. Nested statement – a type of statement inside another statement of the same type; i.e., an If statement inside another If statement, a loop inside another loop
4. Relational operator – an operator that compares data; greater than, less than, equal, greater than or equal, less than or equal, not equal, or the symbols that represent them
5. Selection structure (switch/select case) – a statement selects the correct path depending on the value of a variable
6. Truth table – a table that lists all the possible values of a Boolean expression

Unit 8: Loops

1. Accumulator – a variable that is used to keep a sum of the values of another variable; uses the pattern *variable* = + *number*; i.e., `sum = sum + total`
2. Counter – a variable that is used to keep count of the number of times a loop executes or some other event occurs; uses the pattern *variable* = *variable*; i.e., `number of students = number of students + 1`
3. Nested loop – a loop inside another loop
4. Sentinel loop – a loop that continues until a flag or signal for end of that data is encountered